FAM model

How to:

- Define function name
- Define input parameters
- Define output
- Define list of instructions forming algorithm
  - Level of detail based on modelled problem

Ex: Write function, given input int n, computes factorial of n

```
int computeFactorial(int n)
    for f = 1, n-1, n > 0
        f * n
    return f
```

```
function factorial(n)
    input - n: integer
    output - f : integer

    for(i=n, i>0, --i){
        f = f * i
    }
    return f
```

```
if (n=1)
    return n
else
    return factorial(n-1)*n
```

Random access machine (RAM) model

- Unbounded # of memory cells
- Single CPU
- Computes number of unit times required for running algorithm

Key idea

- Each instruction costs 1
- Count total number of operation

RAM model of factorial using loops

- i = n — 1
- repeat n times:
  - i = i-1 — 2
  - f = f * i — 2
  - i > 0 — 1

1 + 5n

```
f = 1                    1
for(i = n; i > 0; --i)   4n
    f = f * i
return f                 1
```

RAM of factorial using recursion

```
if (n=1)          1
    return n      1
else
    return factorial(n-1)*n   3
```

$2 + 3(n-1)$

$$T(n) = \begin{cases} 2 & \text{if } n = 1 \\ T(n-1) + 3 & \text{if } n > 1 \end{cases}$$

CLEMSON
School of COMPUTING

Review on Algorithms and RAM model

How to write an algorithm?

- Define your function name
- Define your function's input parameters
- Define your function's output
- Define the list of instructions forming your algorithm
  - The level of detail should be set based on the problem you are modeling

How to write an algorithm?

- Write a function that, given an input integer n, computes the factorial of n.

function factorial(n)

input – n: integer

output – f: integer

Option 1 – using loops

```
function factorial(n)
        input – n: integer
        output – f: integer

        f = 1
        for(i = n; i>0; --i ){
                f = f * i
        }
        return f
```

Option 2 – using recursion

```
function factorial(n)
        input – n: integer
        output – f: integer

        if(n = 1)
                return n
        else
                return factorial(n-1)*n
```

Characterize an algorithm's complexity

- Random-access machine (RAM) model
  - Unbounded number of memory cells
  - Single CPU
  - Computes the number of unit times required for running the algorithm

- Key idea
  - Each instruction (arithmetic operation, memory access, et.) costs 1
  - You need to count the total number of operations

RAM model on Factorial (Opt 1)

```
function factorial(n)
        input – n: integer
        output – f: integer

        f = 1                          1
        for(i = n; i>0; --i ){         1
                f = f * i              4n
        }
        return f                       1
```

T(n) = 4n+3

RAM model on Factorial (Opt 2)

```
function factorial(n)
        input – n: integer
        output – f: integer

        if(n = 1)                          1
                return n                   1
        else
                return factorial(n-1)*n    1 1 1
```

$$T(n) = \begin{cases} 2 & if\ n = 1 \\ T(n-1) + 3 & if\ n > 1 \end{cases}$$

What's next?

- Book chapters
  - Chapter 1.1 (from 1.1.1 to 1.1.4)
- In class activities
  - Use the RAM model on new algorithms
- Next class
  - Review on asymptotic notation

Exercise

The Fibonacci numbers are the numbers in the following integer sequence.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

In mathematical terms, the sequence F_n of Fibonacci numbers is defined by the recurrence relation: $F_n = F_{n-1} + F_{n-2}$ with seed values: $F_0 = 0$ and $F_1 = 1$.

Write an algorithm to get the n-th Fibonacci number (n > 0).

Option 1 - Use Loops

```
function fibonacci(n)
     input - n: integer
     output - n: integer


     int arr[n+1];
     arr[0] = 0;
     arr[1] = 1;
     for (i = 2; i <= n; i++)
          arr[i] = arr[i-1] + arr[i-2];
     return arr[n];
```

Option 2 - Use Recursion

```
function fibonacci(n)
     input - n: integer
     output - n: integer

     if (n < 2)
          return n;
     else
          return fibonacci(n-1) + fibonacci(n-2);
```

Use RAM Model on Two Algorithms (1st)

```
function fibonacci(n)
     input - n: integer
     output - n: integer

     int arr[n+1];
     arr[0] = 0;
     arr[1] = 1;
     for (i = 2; i <= n; i++)
          arr[i] = arr[i-1] + arr[i-2];
     return arr[n];
```

Use RAM Model on Two Algorithms (1st)

```
function fibonacci(n)
     input - n: integer
     output - n: integer

     int arr[n+1];                          1
     arr[0] = 0;                            1
     arr[1] = 1;                            1
     for (i = 2; i <= n; i++)               1 (assign)
          arr[i] = arr[i-1] + arr[i-2];     1(<=) 1 (i-1) 1(i-2) 1(+) 1(=) 1(++)
                                            1 (last comparison when i = n + 1)
     return arr[n];                         1
```

Use RAM Model on Two Algorithms (1st)

```
function fibonacci(n)
     input - n: integer
     output - n: integer

     int arr[n+1];                          1
     arr[0] = 0;                            1
     arr[1] = 1;                            1
     for (i = 2; i <= n; i++)               1
          arr[i] = arr[i-1] + arr[i-2];     6(n-1)
                                            1
     return arr[n];                         1
```

T(n) = 6(n-1) + 6 = 6n

Use RAM Model on Two Algorithms (2nd)

```
function fibonacci(n)
     input - n: integer
     output - n: integer

     if (n < 2)
          return n;
     else
          return fibonacci(n-1) + fibonacci(n-2);
```

Use RAM Model on Two Algorithms (2nd)

```
function fibonacci(n)
     input - n: integer
     output - n: integer

     if (n < 2)                                          1
          return n;                                      1
     else
          return fibonacci(n-1) + fibonacci(n-2);        1 1 1 1
```

T(n) = T(n-1) + T(n-2) + 4

Announcement

- Office Hours:
  - Guoxi Liu (guoxil@clemson.edu)
    - Monday 11 - 12 AM, McAdams Hall 110D
    - Tuesday 2 - 3 PM, McAdams Hall 109
  - Xueyi Bao (xueyib@clemson.edu)
    - Wednesday 11 - 12 AM, McAdams Hall 110D
    - Thursday 1 - 2 PM, McAdams Hall 110B
  - Or schedule an appointment with Zoom
- Join the slack channel!